Simulation of Quantum Algorithms for Solving Machine Learning Problems

Steven Oud
SURFsara, Amsterdam

November 20, 2019

Summary

Quantum computing is an emerging technology of the near future promising to solve certain problems intractable by classical computers. Recently, research towards its application in the field of machine learning has shown promise. We take a look at how to apply current state-of-the-art quantum algorithms to solve machine learning problems. Through quantum circuit simulation, we implement a variational quantum neural network for classifying handwritten digits and demonstrate that our QNN can be trained to perform binary, one-vs-rest and multiclass classification. We then demonstrate the QNN's ability to perform various other supervised classification and regression tasks. Given its variational nature, the QNN is aimed at near-term quantum processors capable of only executing low-depth circuits and consisting of up to a few hundred noisy qubits.

Contents

1 Introduction **3**

2 Methods **3**

3 Quantum Computation and Information **4**
- 3.1 Quantum States . . . 4
- 3.2 Quantum State Evolution . . . 5
- 3.3 Measurement . . . 7
- 3.4 Quantum Speedup . . . 8
- 3.5 Current State of Quantum Computing . . . 8

4 Classical Machine Learning **9**

5 Quantum Machine Learning **9**

6 Implementation and Results **10**
- 6.1 Quantum Neural Network . . . 11
  - 6.1.1 Binary Downsampled Classifier . . . 11
  - 6.1.2 One-vs-Rest Classifier . . . 13
  - 6.1.3 Multiclass Classifier . . . 13
  - 6.1.4 Regression Results . . . 14
  - 6.1.5 Discussion and Further Work . . . 16

7 Conclusion **16**

References **17**

Appendices **20**

A Classification Results on Other Data **20**

B CPU and GPU Simulation Benchmarks **21**

1 Introduction

Quantum computing is one of the most promising technology developments of the coming years. Big companies like IBM (IBM, 2019), Google (Google AI, 2019), Intel (Intel, 2019), Microsoft (Microsoft, 2019), and countries like the USA and China are investing huge amounts of money into the development of quantum computers (Raymer and Monroe, 2019; Kania and Costello, 2017). The development of practical quantum computers that can be used to solve real-world problems is driven by the expectation that for certain tasks, quantum computers can dramatically outperform classical computers (Preskill, 2012).

Already last year, SURFsara started several activities and collaborations in the field of quantum computing. The overall objective of SURFsara is to support Dutch researchers in taking an early and competitive advantage in quantum computing technologies and facilities while these become available. Like with many emerging compute technologies, it needs early adopters in the scientific computing community to identify problems of practical interest that are suitable as proof-of-concept applications. In the context of the SURF Open Innovation Lab, SURFsara seeks to stimulate and support these advances in scientific research in close collaboration with research groups. Within this context SURFsara is interested in testing, benchmarking and creating good examples of quantum computing applications that can be used by the scientific community.

SURFsara HPC center supports various research institutes and universities of the Netherlands. The chemistry community is currently one of the largest; the machine learning community probably the fastest growing. Both of these fields are large areas of research within the quantum computing field, expecting improvements to be gained from them. As the main interest of SURFsara is to support the potential main use cases of quantum computers in the future, the tasks of this internship will be focused on quantum machine learning (QML) and quantum chemistry (QC). The first part of the internship will be focused on QML, and so will the research in this report.

This report will try to answer the question "How can quantum algorithms be implemented using simulated quantum circuits to solve machine learning problems?" This is further expanded into the following sub-questions:

1. What are current promising QML algorithms?
2. What simulators are best suited for simulating QML quantum circuits?
3. How can these quantum algorithms be integrated into current classical workflow of machine learning applications?
4. How do these quantum algorithms compare to classical algorithms?

This report is structured as follows. First, in Section 2, we describe how the research was conducted. Second, in Section 3, a short introduction to quantum computation and information is given. Third, in Sections 4 and 5, an overview of the classical and quantum machine learning field is given. Finally, in Section 6, a quantum neural network is implemented using quantum circuit simulation and the results are discussed. The conclusion of the research is described in Section 7.

2 Methods

The research in this report was done mostly through desk research, with the help and advice of my colleagues. The first step was to get familiar with the QML field; what background knowledge is required, what is the current state of research and what are the next steps? To get started, several papers in promising areas of research were provided by colleagues. From there, databases such as Google Scholar and arXiv were used to find further information about these areas. Search terms used include *quantum machine learning*, *quantum neural network*, *quantum support vector machine*, *variational quantum eigensolver*, *quantum phase estimation*, *quantum perceptron*, *quantum classifier*, *quantum machine learning simulation*, *quantum Boltzmann machine* and *hybrid quantum optimization*.

3 Quantum Computation and Information

Quantum computers take advantage of quantum mechanical effects such as superposition and entanglement to solve certain problems faster than classical computers. The idea of a quantum computer was first proposed by Richard Feynman for solving problems in physics and chemistry, remarking that "If you want to make a simulation of nature, you'd better make it quantum mechanical, and by golly it's a wonderful problem, because it doesn't look so easy." (Feynman, 1982). Since then the field has advanced a lot with algorithms such as Shor's algorithm for factoring integers (Shor, 1999) and Grover's search algorithm (Grover, 1996). These quantum algorithms show efficient solutions for problems that are considered hard for classical computers.

This section summarizes the basic concepts of quantum computation and information theory. It is a very high-level overview of quantum computation and is far from a complete introduction to the field. For a more complete introduction to quantum computation and information, we refer to the book by Nielsen and Chuang (2009).

3.1 Quantum States

The basic and smallest unit of quantum information is the quantum bit, or qubit. A qubit is a two-state quantum-mechanical system, which can be represented for example in the spin of an electron (but any quantized physical property can be used). A qubit can be in a state of an orthonormal basis $\{|0\rangle, |1\rangle\}$ (corresponding to spin up and spin down for the electron example), but it can also be in a linear combination, or superposition of states. The state of a qubit (the so-called wave function) can be described as following:

$$|\psi\rangle = \alpha_0 |0\rangle + \alpha_1 |1\rangle . \tag{1}$$

Quantum states are denoted using Dirac notation $|\cdot\rangle$ (also called a ket), which is a column vector in $\mathbb{C}^n$. Here the coefficients $\{\alpha_0, \alpha_1\}$ are complex-valued numbers called probability amplitudes. The computational basis states $|0\rangle$ and $|1\rangle$ are defined as the column vectors $(1 \quad 0)^T$ and $(0 \quad 1)^T$ respectively. When we measure a quantum state, it collapses probabilistically to one of the basis states. The absolute square $|\alpha_0|^2$ gives us the probability of finding the particle in state $|0\rangle$, and $|\alpha_1|^2$ gives us the probability of finding the particle in state $|1\rangle$. As we are dealing with probabilities, the quantum state should be normalized: $\sum_{i=0} |\alpha_i|^2 = 1$. For example, consider the following arbitrary state:

$$|\psi\rangle = \frac{1}{\sqrt{2}} |0\rangle - \frac{i}{\sqrt{2}} |1\rangle . \tag{2}$$

After measuring this state, we have a $|1/\sqrt{2}|^2 = 1/2$ chance of the system being in the state $|0\rangle$, and a $|-i/\sqrt{2}|^2 = 1/2$ chance of being in the state $|1\rangle$.

The laws of quantum mechanics greatly restrict our access to information stored in quantum states. We cannot access the amplitudes of a state, other than by preparing and sampling a state many times to get an approximation. When we measure a state, it collapses to a basis state $|j\rangle$ with probability $|\alpha_j|^2$. If we measure again immediately after the first measure, we get the same result. So if we measure $|1\rangle$ and measure again immediately after, we will see $|1\rangle$ again. Measuring a quantum state collapses the wave function and destroys state information.

A multi-qubit system with n states $\{|\psi_1\rangle, |\psi_2\rangle, \ldots, |\psi_n\rangle\}$ can be represented using the Kronecker product:[1]

$$|\Psi\rangle = |\psi_1\rangle \otimes |\psi_2\rangle \otimes \cdots \otimes |\psi_n\rangle , \tag{3}$$

resulting in a $N = 2^n$ dimensional state $|\Psi\rangle$. For example, a three qubit state lives in a 2^3-dimensional Hilbert space spanned by computational basis states $\{|000\rangle, |001\rangle, |010\rangle, \ldots, |111\rangle\}$:

[1] Entangled states are an exception to this, which will be further discussed in Section 3.2

$$
\begin{aligned}
|\Psi\rangle &= \alpha_0 |000\rangle + \alpha_1 |001\rangle + \alpha_2 |010\rangle + \cdots + \alpha_{N-1} |111\rangle \\
&= \alpha_0 \begin{pmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \alpha_1 \begin{pmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} + \alpha_2 \begin{pmatrix} 0 \\ 0 \\ 1 \\ \vdots \\ 0 \end{pmatrix} + \cdots + \alpha_{N-1} \begin{pmatrix} 0 \\ 0 \\ 0 \\ \vdots \\ 1 \end{pmatrix} = \begin{pmatrix} \alpha_0 \\ \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_{N-1} \end{pmatrix}.
\end{aligned} \tag{4}
$$

Note that the state space of a quantum state grows exponentially with the number of qubits. This is where some of the potential power of quantum computers comes from, we need 2^n complex numbers to describe an n qubit state.

3.2 Quantum State Evolution

In classical computers, we manipulate bits through logical gates. Equivalently, quantum computers use quantum gates, which transforms one quantum state to another through an operator U. These operators must be unitary. That is, they must preserve the norm of the vector and be reversible: $U^\dagger U = UU^\dagger = I$ (where † is the complex conjugate and I the identity matrix). Single qubit states can be thought of as a vector on the surface of a sphere (the Bloch sphere). A unitary operation can then be thought of as rotations around the x, y and z axes of the sphere (Figure 1). For bigger and more complex quantum operations we will often use circuit notation, as shown in Figure 2.

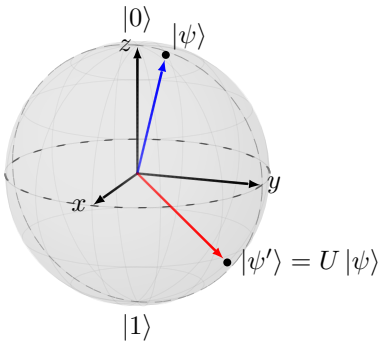


Figure 1: Arbitrary transformation of state $|\psi\rangle$ by operator U visualized on the Bloch sphere.

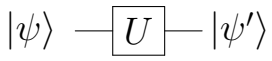


Figure 2: Quantum circuit notation of the operation $|\psi'\rangle = U|\psi\rangle$.

As an example, the X gate is equivalent to the classical NOT: $X|0\rangle = |1\rangle$ and $X|1\rangle = |0\rangle$. It is known as one of the three Pauli matrices used in quantum mechanics:

$$
\sigma_x = X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad \sigma_y = Y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad \sigma_z = Z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{5}
$$

Another frequently used gate is the Hadamard gate, which produces an equal superposition:

$$
H = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{6}
$$

It acts on the computational basis states as following:

$$
\begin{aligned}
H|0\rangle &= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \begin{pmatrix} 1 \\ 0 \end{pmatrix} \\
&= \frac{1}{\sqrt{2}} \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\
&= \frac{1}{\sqrt{2}} (|0\rangle + |1\rangle),
\end{aligned} \tag{7}
$$

$$
\begin{aligned}
H|1\rangle &= \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 1\\ 1 & -1\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix} \\
&= \frac{1}{\sqrt{2}}\begin{pmatrix}1\\-1\end{pmatrix} \\
&= \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle).
\end{aligned} \tag{8}
$$

Applying the Hadamard gate again on the equal superposition created above is an example of destructive interference:

$$H\frac{1}{\sqrt{2}}(|0\rangle + |1\rangle) = |0\rangle\,, \tag{9}$$

$$H\frac{1}{\sqrt{2}}(|0\rangle - |1\rangle) = |1\rangle\,. \tag{10}$$

We find that applying the Hadamard twice is the same as doing nothing, or more formally, we say the Hadamard gate is Hermitian: $H = H^\dagger$.

Multi-qubit gates act on two or more qubits and are required for universal quantum computing. The CNOT (controlled NOT) gate is a quantum gate comparable to a classical XOR gate. It acts on two qubits: a control qubit, and a target qubit. If the control is set to 0, the target qubit is left alone. If the control is set to 1, the target qubit is flipped.

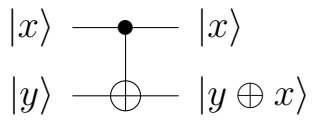


Figure 3: CNOT gate acting on a state $|x, y\rangle$. $\text{CNOT}\,|x, y\rangle = |x, y \oplus x\rangle$, where $\oplus$ represents the binary sum (XOR).

The circuit notation of the CNOT gate is shown in Figure 3. Multi-qubit gates are required for universal quantum computing and introduce the quantum phenomena entanglement. Consider the following state:

$$\left|\Phi^+\right\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \tag{11}$$

Measuring the first qubit gives state $|0\rangle$ with probability 1/2, and state $|1\rangle$ with probability 1/2. However, the measurement immediately collapses the whole state to either $|00\rangle$ or $|11\rangle$. So by measuring the first qubit, we also know the state of the second qubit and vice versa. The individual states are related to each other, and this relation is called entanglement. More formally, we say two qubits are entangled if and only if the state of those two qubits cannot be expressed as two individual states. For the entangled state $|\Phi^+\rangle$, let's assume there exist amplitudes $\{a, b, c, d\}$ such that:

$$\left|\Phi^+\right\rangle = (a\,|0\rangle + b\,|1\rangle) \otimes (c\,|0\rangle + d\,|1\rangle) = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle). \tag{12}$$

This would imply that $\left(ac = bd = 1/\sqrt{2}\right) \wedge \left(ad = bc = 0\right)$, which is a contradiction. Thus, $|\Phi^+\rangle$ is an entangled state.

Another remarkable thing about entanglement is that it works over any distance. That is, given the state $|\Phi^+\rangle$, one qubit could be located in another galaxy while the other qubit is located on earth. When we measure the qubit on earth, we immediately know that the qubit in the other galaxy is in the same state as we measured. This is part of what made Einstein dissatisfied with the theory of quantum mechanics, referring to entanglement as "spooky action at a distance." He argued that it allowed for faster than light communication, thus violating the speed limit on the transmission of information as described in his theory of relativity. However, this is not the case, as a classical channel is still required to communicate the results between the observers.

3.3 Measurement

We can define measurement more systematically. The probability of finding a state $|\psi\rangle$ in state $|\varphi\rangle$ after measurement is given by

$$|\langle\varphi|\psi\rangle|^2, \tag{13}$$

where the notation $\langle\varphi|$, called a bra, represents the conjugate transpose of $|\varphi\rangle$. For example, given a generic quantum state $|\varphi\rangle = (\alpha_0 \quad \alpha_1)^T$, then

$$\langle\varphi| = |\varphi\rangle^\dagger = \begin{pmatrix}\alpha_0\\ \alpha_1\end{pmatrix}^\dagger = \begin{pmatrix}\alpha_0^* & \alpha_1^*\end{pmatrix}. \tag{14}$$

The bra-ket notation $\langle\varphi|\psi\rangle$ can then be thought of as the inner product between the vectors $|\varphi\rangle$ and $|\psi\rangle$. Recall the example state from Equation 2: $|\psi\rangle = 1/\sqrt{2}\,|0\rangle - i/\sqrt{2}\,|1\rangle$. The probability of measuring $|0\rangle$ can be calculated as follows. First we take the inner product between $|0\rangle$ and $|\psi\rangle$:

$$\begin{aligned}\langle 0|\psi\rangle &= \langle 0|\,\frac{1}{\sqrt{2}}\,|0\rangle - \langle 0|\,\frac{i}{\sqrt{2}}\,|1\rangle\\ &= \frac{1}{\sqrt{2}}\,\langle 0|0\rangle - \frac{i}{\sqrt{2}}\,\langle 0|1\rangle\\ &= \frac{1}{\sqrt{2}}\begin{pmatrix}1 & 0\end{pmatrix}\begin{pmatrix}1\\0\end{pmatrix} - \frac{i}{\sqrt{2}}\begin{pmatrix}1 & 0\end{pmatrix}\begin{pmatrix}0\\1\end{pmatrix}\\ &= \frac{1}{\sqrt{2}}.\end{aligned} \tag{15}$$

Then we take the absolute square to give

$$|\langle 0|\psi\rangle|^2 = \left|\frac{1}{\sqrt{2}}\right|^2 = \frac{1}{2}, \tag{16}$$

yielding the result we expected. The measurement $|\langle\varphi|\psi\rangle|^2$ is also sometimes referred to as the overlap or fidelity between $|\psi\rangle$ and $|\varphi\rangle$, as it essentially measures the closeness of two quantum states. The orthogonality of the computational basis states can be understood from this: $\langle 0|0\rangle = \langle 1|1\rangle = 1$ and $\langle 0|1\rangle = \langle 1|0\rangle = 0$.

When we measure a quantum state, we measure with respect to a basis. So far we have been describing measurement in the computational basis, which for a single qubit consists of the basis states $\{|0\rangle, |1\rangle\}$. However, the choice of basis is arbitrary and can be described more generally. We can describe a measurement by an observable O, which is a Hermitian operator. When we measure an observable O, we measure one of its eigenvalues λ_i, and the state collapses to the associated eigenvector $|\lambda_i\rangle$. In the computational basis, we measure the Pauli Z operator σ_z (Equation 5), which has eigenvalues $\{1, -1\}$ with eigenvectors $\{|0\rangle, |1\rangle\}$.

We are often interested in the expectation value of an observable for a state $|\psi\rangle$. The expectation value $\langle O\rangle$ of an observable O can be thought of as an average of all the possible measurement outcomes (the eigenvalues of O), weighted by their probability. It can be calculated as following:

$$\begin{aligned}\langle O\rangle &= \langle\psi|\,O\,|\psi\rangle\\ &= \left(\alpha_0^*\,\langle\lambda_0| + \ldots + \alpha_{N-1}^*\,\langle\lambda_{N-1}|\right)\left(\alpha_0 O\,|\lambda_0\rangle + \ldots + \alpha_{N-1} O\,|\lambda_{N-1}\rangle\right)\\ &= \left(\alpha_0^*\,\langle\lambda_0| + \ldots + \alpha_{N-1}^*\,\langle\lambda_{N-1}|\right)\left(\alpha_0\lambda_0\,|\lambda_0\rangle + \ldots + \alpha_{N-1}\lambda_{N-1}\,|\lambda_{N-1}\rangle\right)\\ &= |\alpha_0|^2\,\lambda_0 + \ldots + |\alpha_{N-1}|^2\,\lambda_{N-1}\\ &= \sum_{i=0}^{N-1} |\alpha_i|^2\,\lambda_i.\end{aligned} \tag{17}$$

Experimentally, this would mean doing repeated measurements until we get a good approximation of $\langle O\rangle$. Note that repeated measurements means repeated state preparation and measurement, as the state collapses after measurement.

3.4 Quantum Speedup

Certain quantum algorithms promise to provide a speedup over classical algorithms by "exploiting" quantum phenomena such as superposition, interference, and entanglement (Nielsen and Chuang, 2009). They usually offer an exponential or polynomial speedup over their classical counterpart. Scott Aaronson probably described it best: "The goal in quantum computing is to choreograph a computation so that the amplitudes leading to wrong answers cancel each other out, while the amplitudes leading to right answers reinforce." (Aaronson, 2011). It is far from obvious how to design a quantum algorithm, especially one that provides a speedup over classical algorithms. Quantum algorithm design requires a completely different way of thinking compared to classical algorithm design and requires interdisciplinary knowledge of mathematics, computer science, and physics.

The most famous example of a quantum speedup is probably Shor's algorithm for factoring integers and computing discrete logarithms, which was introduced in 1994. Its implications are huge for modern cryptography algorithms such as RSA, which depend on the fact that factoring integers is a hard problem on classical computers. Shor showed that this can be done in polynomial time on a quantum computer (Shor, 1999), providing an exponential speedup over classical algorithms. It has been estimated that a 2048-bit RSA integer could be factored in eight hours using 20 million noisy qubits (Gidney and Ekerå, 2019).

There are other quantum algorithms that promise a significant speedup, such as Grover's algorithms for searching unstructured databases (Grover, 1996) and the quantum phase estimation algorithms for finding eigenvalues of a unitary operator (Nielsen and Chuang, 2009). However, these algorithms often require quantum computers with large amounts of coherent qubits, which we will most likely not have access to any time soon.

3.5 Current State of Quantum Computing

It is important to note that quantum computers are still at the experimental stage. For example, the highest number factored by using Shor's algorithm on an actual quantum computer thus far is 21 (Martín-López et al., 2012). Experimentally proving that quantum computers can do something that classical computers cannot do efficiently is an important milestone to reach. The term quantum supremacy was coined by Preskill (2012) to describe this achievement. It was estimated that current or near-future quantum computers—often referred to as noisy intermediate-scale quantum (NISQ) devices (Preskill, 2018)—which have about 50-100 qubits, may be able to achieve quantum supremacy (Boixo et al., 2018).

Very recently Google claimed to have achieved quantum supremacy with 53 qubits by doing random circuit sampling many times faster on a quantum computer than a classical computer (Arute et al., 2019). They claim to be able to perform the computation in 200 seconds on a quantum computer and determined it would take the world's fastest supercomputer 10,000 years to perform a similar computation. IBM responded to Google's results by claiming they could simulate the Google experiment in 2.5 days, rather than the 10,000 years that Google estimated (Pednault, Gunnels, Nannicini, Horesh, and Wisnieff, 2019). While this seems like it would diminish Google's claim of quantum supremacy, 200 seconds versus 2.5 days is still a speedup by a factor of more than a thousand. Even more, increasing the number of qubits from 53 to 55 would be enough to refute IBM's response, as it would exceed the world's most powerful supercomputer's storage capacity (Aaronson, 2019). Either way, Google has made great progress by developing a high-fidelity 53 qubit quantum computer, which will further stimulate the development of NISQ devices.

Research has adapted to the limitations of NISQ devices, and hybrid quantum-classical algorithms have become an important area of research (McClean, Romero, Babbush, and Aspuru-Guzik, 2016; Guerreschi and Smelyanskiy, 2017). These hybrid algorithms are often referred to as variational quantum algorithms. Examples of such algorithms are the quantum approximate optimization algorithm (QAOA) for optimization problems (Farhi, Goldstone, and Gutmann, 2014), and the variational quantum eigensolver (VQE) for finding eigenvalues of a Hamiltonian (Peruzzo et al., 2014). These variational algorithms typically involve a small quantum subroutine run inside

of a classical optimization loop, removing the need for a large-scale, coherent quantum computer.

So far we have been talking about so-called universal gate model quantum computers. There exist special-purpose quantum computers called quantum annealers, which have achieved qubit counts up to 2000 (Gibney, 2017). These quantum annealers are restricted to solving optimization problems, of which its promised speedup has not been clearly shown (Shin, Smith, Smolin, and Vazirani, 2014; Aaronson, 2013; Ronnow et al., 2014). This research is focused on universal gate model quantum computers, as its usefulness and applications appear more evident.

4 Classical Machine Learning

Before looking at quantum machine learning, we first explore the field of classical machine learning, which quantum machine learning is largely built upon. Instead of being explicitly programmed to perform a certain task, machine learning algorithms learns by analyzing large amounts of data. As a subfield of artificial intelligence, machine learning is used to perform complex tasks that come naturally to us humans, such as image recognition, spam detection, and recommendation.

Machine learning algorithms can be broadly divided into three categories: *supervised*, *unsupervised* and *reinforcement learning*. In supervised learning, we have a predefined set of training data in which data points are correctly labeled. Its goal is then to predict the correct label for new data it has not seen. In unsupervised learning, we do not have labeled data points, and instead, we try to find previously unknown patterns in the data set. Finally, in reinforcement learning, an agent learns how to behave in an environment by interacting with it and receiving feedback based on these interactions.

Artificial neural networks (ANN) are ubiquitous in the field of supervised machine learning. They are inspired by the biological neural networks found in our brains and have developed into a wide range of methods that have advanced the state of the art in various fields. Some of the methods include feedforward neural networks for general classification and regression, convolutional neural networks for processing visual data (Ciresan, Meier, and Schmidhuber, 2012) and generative adversarial networks for generating new data based on existing data (Goodfellow et al., 2014). Another notable machine learning model for supervised learning is support vector machine (SVM), which can be used for classification or regression. They have been found to be able to solve problems such as semantic parsing (Pradhan, Ward, Hacioglu, Martin, and Jurafsky, 2004) and handwritten character recognition (Decoste and Schölkopf, 2002).

A popular method of unsupervised learning is clustering. Clustering is the task of grouping data points together with other data points that are most similar. Some well-known clustering algorithms include k-means clustering and hierarchical clustering. Generative adversarial networks can be thought of as unsupervised learning algorithms and can even be used in reinforcement learning (Ho and Ermon, 2016).

Classical machine learning algorithms are well defined and can be used to classify images, recognize patterns and speech, play complex games and much more. However, as the amount of globally stored data is growing by about 20% every year (Hilbert and Lopez, 2011), researchers are looking more and more for innovative approaches to machine learning. A promising idea that is currently being investigated is the use of quantum computers to improve classical machine learning algorithms (Schuld, Sinayskiy, and Petruccione, 2014). The research field of quantum machine learning may offer new ways of intelligent data processing and even guide and advance classical machine learning.

5 Quantum Machine Learning

It has been known for a while that for some problems quantum algorithms can offer a speedup over their classical counterpart (Nielsen and Chuang, 2009). Recent advances in quantum computing suggest that machine learning algorithms may also be susceptible to a quantum speedup (Lee et al., 2019; Lloyd, Mohseni, and Rebentrost, 2013; Gao, Zhang, and Duan, 2018; Yoo, Bang, Lee,

and Lee, 2014; Biamonte et al., 2017). Quantum machine learning is still in its very first stage of development and lacks a widely agreed-upon definition. However, there have been promising developments in recent years. An efficient quantum algorithm named HHL (after its inventors) for solving a system of linear equations was found by Harrow, Hassidim, and Lloyd (2009). In the years since, HHL has been proposed for multiple machine learning applications, such as k-means clustering (Lloyd et al., 2013), linear regression (Schuld, Sinayskiy, and Petruccione, 2016; Yu, Gao, and Wen, 2017; Wang, 2017), and support vector machines (Rebentrost, Mohseni, and Lloyd, 2014).[2]

HHL-based algorithms require quantum computers capable of executing high-depth circuits, which we will not have access to any time soon. To circumvent this requirement, variational QML algorithms have become a main focus area of research. Variational QML algorithms have shown the ability to perform machine learning tasks on NISQ devices while providing a means to deal with high dimensional data, which has been unpractical on classical computers (Mitarai, Negoro, Kitagawa, and Fujii, 2018). Promising results have been shown with quantum neural networks (Farhi and Neven, 2018; Schuld, Bocharov, Svore, and Wiebe, 2018; Grant et al., 2018), quantum generative adversarial networks (Romero and Aspuru-Guzik, 2019; Benedetti, Grant, Wossnig, and Severini, 2019), quantum support vector machines (Havlíček et al., 2019; Schuld and Killoran, 2019; Ghobadi, Oberoi, and Zahedinejhad, 2019), clustering (Otterbach et al., 2017) and quantum Boltzmann machines (Verdon, Broughton, and Biamonte, 2017; Anschuetz and Cao, 2019).

Quantum machine learning can help us process the huge amounts of data we keep creating. Furthermore, they can potentially speed up certain computational heavy parts used in machine learning algorithms by a large factor. Finally, we can use QML algorithms to learn properties about quantum data, for which exists no classical equivalent. However, to realize these applications we need to overcome some major obstacles. First, we need to build quantum computers with more qubits, better qubits and longer coherence times. Second, we need an efficient way to prepare classical data into a quantum state. A quantum random access memory (qRAM) architecture has been proposed for this (Giovannetti, Lloyd, and Maccone, 2008), but it largely depends on the data being relatively uniform (Aaronson, 2015). Third, similarly to the previous point, is the readout problem. Since measurement is probabilistic and destroys a quantum mechanical system's wave function, we need to do repeated measurements to extract any meaningful information of the system. So the output could be contained in an exponentially large Hilbert space, but we cannot efficiently extract this information.

6 Implementation and Results

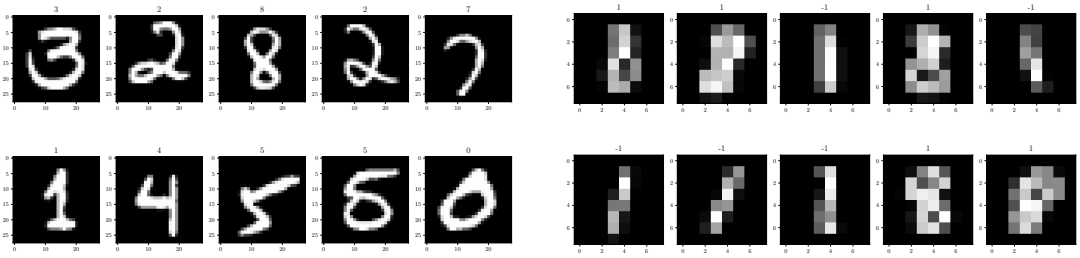
This section describes the experiments conducted, the results and possible future work. While we would eventually want to run algorithms on actual quantum hardware, we will focus on simulation because of the limitations of current quantum hardware. The experimental results described were obtained by simulating quantum circuits using SURFsara's computing systems (SURFsara, 2019). Simulations were done using the PennyLane (Bergholm, Izaac, Schuld, Gogolin, and Killoran, 2018) framework. For performance reasons, a PennyLane back-end for the Qulacs quantum circuit simulator (Qulacs, 2019) was implemented, effectively speeding up simulations by a factor of 2 and providing GPU support.[3] We benchmark and compare CPU and GPU simulation run times with Qulacs in Appendix B.

[2] HHL and the algorithms which use it can only provide an exponential speedup by overcoming some major caveats (Aaronson, 2015). For example, the problem of loading a large amount of classical data into a quantum computer can quickly diminish or destroy the potential quantum speedup.

[3] Available at https://github.com/soudy/pennylane-qulacs/.

6.1 Quantum Neural Network

A quantum neural network (QNN) inspired by the work of Farhi and Neven (2018) and Pérez-Salinas, Cervera-Lierta, Gil-Fuster, and Latorre (2019) was implemented. The MNIST database of handwritten digits (LeCun and Cortes, 2010) was used as data set (Figure 4) to demonstrate its ability to do classification. This data set is often used in introductions to machine learning, so it seems natural to use this data set to test our QNN. For the initial experiment, the images were downsampled to a lower dimension and the task at hand was limited to binary classification of digits 1 and 8 to simplify the problem and reduce resource requirements. After a successful initial experiment, the experiment was extended to one-vs-rest classification of the original 28×28 data set, and finally generalized to multi-class classification of the original data set. In Section 6.1.4 we also demonstrate the QNN's ability to do simple regression tasks.



(a) Original data set of 28×28 images. **(b)** Downsampled binary data set of 8×8 images.

Figure 4: Samples of the MNIST handwritten digits data set used for classification.

6.1.1 Binary Downsampled Classifier

The problem of supervised classification in machine learning is defined as follows: given a training set of data whose category is known, train a model which is able to identify to which category new data belongs. A simple example that we will tackle in this section is recognizing if a handwritten digit is a 1 or an 8.

We implement a variational QNN to classify 8×8 images of handwritten digits (Figure 4b). It uses a quantum circuit together with a classical optimization algorithm to find the optimal parameters for the quantum circuit. An overview of the quantum circuit used in the experiment can be found in Figure 5. The variational QNN classifier learns as follows:

1. Prepare data sample $\vec{x}$ in the amplitudes of a quantum state $|x\rangle$:

$$|x\rangle = \frac{1}{\|\vec{x}\|} \sum_{i=0}^{N-1} x_i \, |i\rangle \,, \tag{18}$$

   where $|i\rangle$ represents the i'th computational basis state.

2. Apply L layers of parameterized unitaries: $\mathcal{U}(\vec{\theta})|x\rangle$, where $\mathcal{U}(\vec{\theta}) = U_\ell(\vec{\theta}_L)U_\ell(\vec{\theta}_{L-1}) \dots U_\ell(\vec{\theta}_1)$.

3. Measure expectation value $\langle Z \rangle = \langle x| \, \mathcal{U}(\vec{\theta})^\dagger Z_n \, \mathcal{U}(\vec{\theta}) \, |x\rangle$ of last qubit.

4. Calculate and minimize loss $\mathcal{L}(\vec{\theta})$ through a classical optimization algorithm.

5. Repeat until convergence.

For the binary classifier we use the square loss cost function:

$$\mathcal{L}(\vec{\theta}) = \sum_{i=1}^{D} \Big(y_d - \langle x_d| \, \mathcal{U}(\vec{\theta})^\dagger Z_n \, \mathcal{U}(\vec{\theta}) \, |x_d\rangle \Big)^2 \,, \tag{19}$$

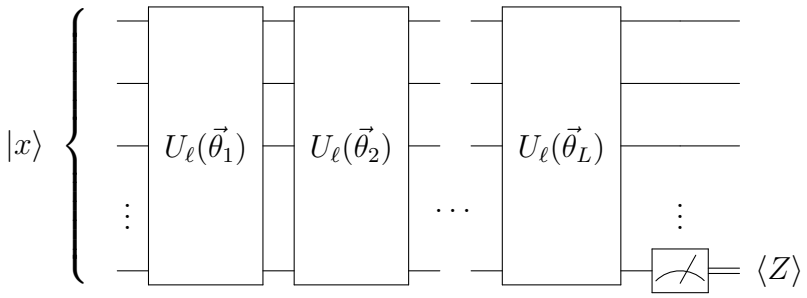


Figure 5: General circuit of the QNN. The network consists of L layers with each layer implementing a parameterized unitary $U_\ell(\vec{\theta})$ whose parameters are optimized using a classical optimization algorithm. The decomposition of a layer is shown in Figure 6.

where D is the number of training samples and y_d the correct label for input x_d. Predicting the label y for input $\vec{x}$ is done by taking the sign of the Z expectation value of the last qubit:

$$y(\vec{x}) = \text{sign}\left(\langle x|\mathcal{U}(\vec{\theta})^\dagger Z_n \mathcal{U}(\vec{\theta})\,|x\rangle\right). \tag{20}$$

We trained a QNN using a training set of 3148 images and a test set of 843 images. As optimizer, we used mini-batch stochastic gradient descent (SGD) with a learning rate of 0.05 and a batch size of 32. Due to computational limits, we restricted the initial experiment to 20 epochs. After training for 20 epochs with 2 layers, it reached 97.2% test accuracy without any hyperparameter tuning or circuit optimization. The goal of this experiment is not to compete with classical state-of-the-art neural networks, which are capable of achieving an error percentage of only 0.23% on multiclass classification of all the digits (Ciresan et al., 2012).[4] Rather, it is meant to demonstrate the possibility of using quantum computers for machine learning problems and to hint that the universal approximation theorem (Csáji, 2001) applies to quantum neural networks. It shows the potential to use quantum computers to process (quantum) data that classical computers cannot process efficiently.

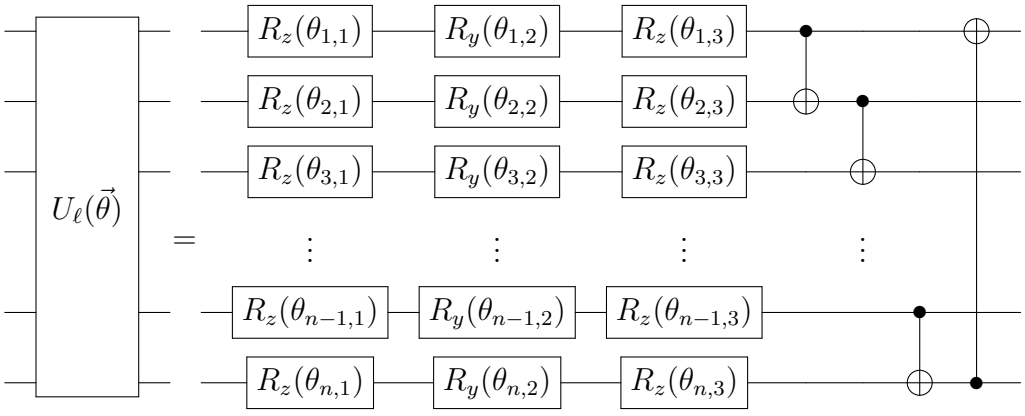


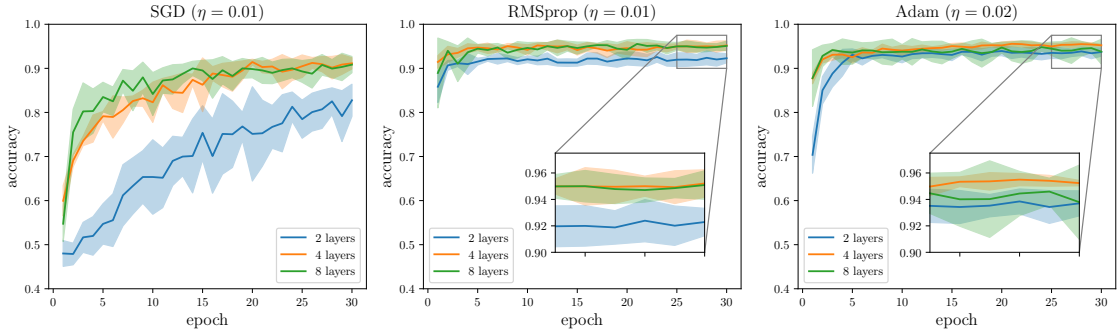
Figure 6: Decomposition of a layer $U_\ell(\vec{\theta})$. The amount of trainable parameters for a QNN with n qubits and L layers is $3n \cdot L$. The number of qubits required is decided by the dimension of the data: $n = \lceil \log_2 N \rceil$, where N is the input dimension. The downsampled MNIST classifier requires $\lceil \log_2(8 \cdot 8) \rceil = 6$ qubits.

[4] Which is not to say quantum neural networks can't compete with classical neural networks. In fact, Pérez-Salinas et al. (2019) show a quantum classifier with two qubits to be at least comparable with classical methods by using data re-uploading techniques.

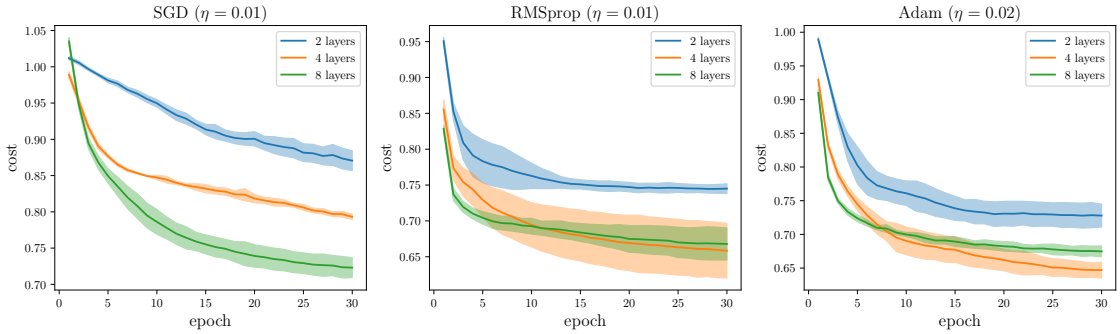
6.1.2 One-vs-Rest Classifier

To make a step into the direction of multi-class classification using a QNN, we first consider the problem of one-vs-rest classification. We use the original 28×28 MNIST data set (Figure 4a) to train a QNN to detect if a handwritten digit is 0 or not. To encode a 28×28-dimensional image we need $\lceil \log_2(28 \cdot 28) \rceil = 10$ qubits. As the complexity of the problem grows, we compare the performance of multiple optimizers: SGD, RMSprop, and Adam. We also compare the performance with 2, 4 and 8 layers to see what impact increasing the number of layers has. We trained the network for 30 epochs with a batch size of 32. The training and testing data set consisted of 710 and 1372 images respectively. Some hyperparameter tuning was done to find the optimal learning rate, but more tuning can be done to possibly increase performance.

The results of the experiment are shown in Figure 7. In general, the performance of the classifiers seems to improve as we increase the number of layers, but ultimately it does not seem to benefit from more than 4 layers. Adaptive learning rate methods RMSprop and Adam convergence faster and find a better optimum compared to vanilla SGD. The amount of trainable parameters per network is 60 for 2 layers, 120 for 4 layers and 240 for 8 layers. In contrast, a classical neural network has at least $28 \cdot 28 = 784$ parameters just for the input layer.



(a) Accuracy on test data set with different optimizers.



(b) Cost landscape of different optimizers.

Figure 7: Experimental results of a one-vs-rest QNN classifier for detecting if a handwritten digit is 0 or not. The data shown is the mean taken from five runs with the standard deviation plotted as the area under the mean.

6.1.3 Multiclass Classifier

Finally we generalize the QNN classifier to a multiclass classifier by implementing a fidelity loss function as proposed by Pérez-Salinas et al. (2019). To implement this we need to make some changes to the QNN described in Section 6.1.1. First, we introduce C maximally orthogonal label states, where C is the number of classes (Figure 8). Then, instead of using the square loss function

on the expectation value as cost, we measure the fidelity of the final state of the last qubit and the label state. The goal then is to maximize the fidelity between the final state of the last qubit and the correct label state. We define the fidelity cost function as following:

$$\mathcal{F}(\vec{\theta}) = -\frac{1}{D}\sum_{d=1}^{D} \log\left(|\langle y_d|\mathcal{U}(\vec{\theta})\,|x_d\rangle|^2\right), \tag{21}$$

where D is the number of training samples, $|y_d\rangle$ the correct label state of input $|x_d\rangle$, and $\mathcal{U}(\vec{\theta})$ the parameterized layer unitaries. To make a prediction, we measure the fidelities between the output and all the label states and choose the label with the highest fidelity.

We conduct a similar experiment to that from the previous section. The problem at hand is to distinguish between the handwritten digits 1 through 6 including, giving six classes. We take the six polar states $\{|0\rangle, |1\rangle, |+\rangle, |-\rangle, |+i\rangle, |-i\rangle\}$ as our maximally orthogonal label states (Figure 8). The network was shown the training data set consisting of 27,009 samples once and was tested on 6009 samples. The results are shown in Table 1. All optimizers seem to perform relatively equal, with the adaptive optimizers finding a slightly better optimum with 10 layers. We see a bigger performance increase when increasing the number of layers than before, likely due to the task at hand being more complex. Due to the computational limits of quantum circuit simulation, the network was shown the dataset for only one epoch. We predict the QNN's performance can be improved with more epochs, by redundantly encoding data and by using data re-uploading techniques as suggested by Vidal and Theis (2019) and Pérez-Salinas et al. (2019).

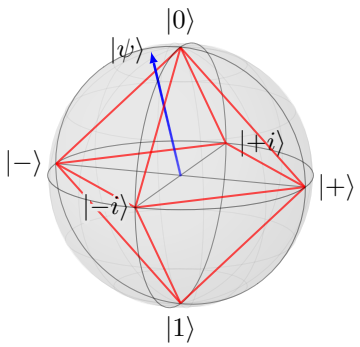


Figure 8: Bloch sphere representation of six maximally orthogonal points which are used as labels. The circuit output state $|\psi\rangle$ would get classified to the class corresponding to the label state $|0\rangle$, as their fidelity $|\langle 0|\psi\rangle|^2$ is the largest.

Optimizer	Layers	Test accuracy	σ
SGD ($\eta = 0.4$)	2	44.6%	0.019
	4	67.8%	0.026
	8	70.7%	0.009
	10	72.6%	0.026
RMSprop ($\eta = 0.01$)	2	49.5%	0.039
	4	66.8%	0.023
	8	70.8%	0.038
	10	74.9%	0.012
Adam ($\eta = 0.02$)	2	45.9%	0.025
	4	68.9%	0.012
	8	71.6%	0.030
	10	74.4%	0.022

Table 1: Quantum neural network performance on multiclass classification of six classes. The accuracy is taken as the mean of five repeated runs.

6.1.4 Regression Results

We take on the problem of learning the functions x^3, $\cos(\pi x)\sin(2\pi x+2)$, $|x|$ and e^x to demonstrate the QNN's ability of representing non-linear real functions. The network is shown 75 samples of the functions in the range $[-1, 1]$ from a uniform distribution and is tested on 100 evenly spaced numbers over the same interval. The data of the functions are one-dimensional, and we employ a data re-uploading technique described by Pérez-Salinas et al. (2019) to achieve generalization with a small number of qubits (Figure 9).

Every layer before applying a parameterized unitary $U_\ell(\vec{\theta})$, we encode a data point x in n qubits

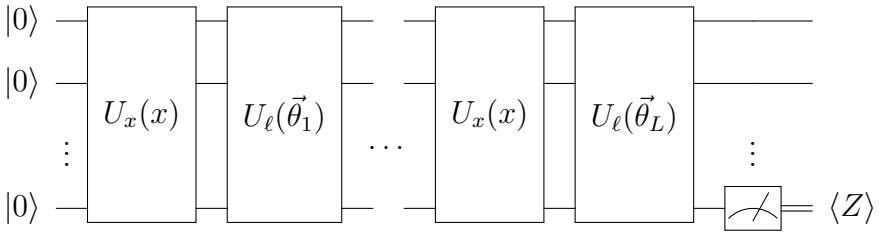


Figure 9: Data re-uploading QNN circuit used to achieve generalization with a small amount of qubits. U_ℓ represents a layer as before in Section 6.1.1, and U_x represents the unitary that encodes the input data in the quantum state. In our example we use one-dimensional data, but U_x can be extended to encode a high-dimensional vector.

as follows:

$$U_x(x) = \prod_{i=1}^{n} R_x^{(i)}(\arccos x) R_y^{(i)}(\arcsin x). \tag{22}$$

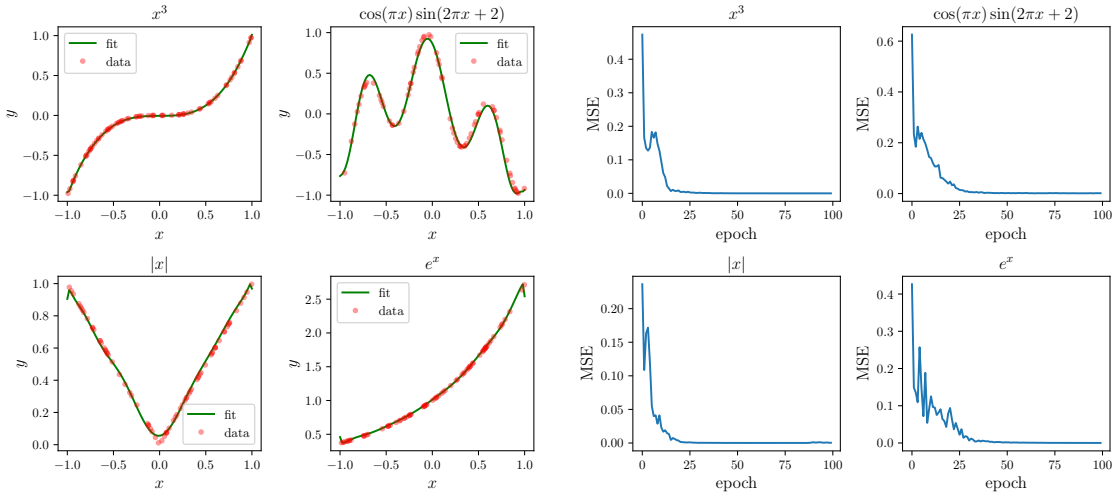
The whole circuit can then be described as

$$\mathcal{U}(\vec{\theta}, x) = U_\ell(\vec{\theta}_L) U_x(x) U_\ell(\vec{\theta}_{L-1}) U_x(x) \dots U_\ell(\vec{\theta}_1) U_x(x), \tag{23}$$

and the network's output is given by

$$y(x) = \gamma \cdot \langle 0| \mathcal{U}(\vec{\theta}, x)^\dagger Z_n \mathcal{U}(\vec{\theta}, x) |0\rangle . \tag{24}$$

Here, γ is a trainable parameter initialized to 1. We use the mean squared error as cost function.



(a) Teacher data vs. learned function. The red points represent the teacher data shown to the network, and the green line represents the learned function.

(b) Landscapes of mean squared error cost function. The QNNs were trained for 100 epochs with 3 layers. RMSprop was used as optimizer with learning rate 0.01 and momentum 0.9.

Figure 10: Performance of our QNN learning continuous functions. Input data was encoded in three qubits according to Equation 22.

The results are shown in Figure 10. We see that the QNN has an easy time learning the wavelike functions x^3 and $\cos(\pi x)\sin(2\pi x+2)$ due to the encoding method. The other functions $|x|$ and e^x are approximated well, but leave room for improvement. Possible improvement can be made by

employing better (problem specific) encoding techniques and more hyperparameter tuning. We have demonstrated a naive approach to doing regression tasks using our QNN. Nevertheless, we managed to train models that approximate these non-linear functions well.

6.1.5 Discussion and Further Work

We have described and implemented a variational QNN capable of classifying classical data and doing simple regression tasks. By applying the QNN to the problem of handwriting recognition, we have demonstrated its capability of solving real-world problems. Our QNN exploits the exponential Hilbert space of quantum states to encode classical data in an amount of qubits logarithmic to the input data's dimension. This demonstrates the possibility to process high dimensional (quantum) data that is intractable for classical computers to process. In Appendix A we apply the QNN classifier on other data sets to show its ability to adapt to various problems.

Further work can be done by using more quantum-specific optimization methods (Stokes, Izaac, Killoran, and Carleo, 2019), optimizing the measuring of expectation values (Sweke et al., 2019), improving the circuit initialization strategy (McClean, Boixo, Smelyanskiy, Babbush, and Neven, 2018), conducting more experiments on quantum and classical data, defining useful benchmarks and experimenting with real quantum hardware. Improvements can also be made to the encoding of classical data into a quantum Hilbert space by experimenting with different feature maps (Schuld and Killoran, 2019).

7 Conclusion

Machine learning is ubiquitous and widely used to perform complex tasks such as image recognition, playing games, recognizing speech and more. Recently, researchers have been looking for ways to apply quantum computers to machine learning, resulting in the field of quantum machine learning. Quantum machine learning is an exciting new field that shows the potential to speed up certain machine learning algorithms by using quantum computers. However, most known QML algorithms require a large amount of quantum resources and cannot be implemented on the NISQ devices we will have access to in the near future. This has caused much research to shift towards the development of variational algorithms, which involve a small quantum subroutine run inside of a classical optimization loop, removing the need for a large-scale, coherent quantum computer. Though it is likely that most QML approaches will not be of much practical use in the near future, variational algorithms might already provide a quantum advantage at the level of 50-100 qubits for certain problems. However, we should not expect the promised exponential speedups to be achieved on near-term devices. In this report we implemented a variational QNN using quantum circuit simulation to demonstrate how quantum computers can be used to solve machine learning problems. We hope this work will somewhat help advance the field of QML and make QML more accessible to researchers around the world. There is still much work to be done in both implementation and theory, but the work done so far is promising. Quantum computing may very well solve some of the computational problems we are facing today and revolutionize the field of machine learning.

References

Aaronson, S. (2011). Quantum computing promises new insights, not just supermachines. In *Quantum Computing since Democritus* (Chap. Quantum, pp. 109–131). Cambridge University Press

Aaronson, S. (2013). D-Wave: Truth finally starts to emerge. Retrieved 2019-09-19, from https://www.scottaaronson.com/blog/?p=1400

Aaronson, S. (2015). "Read the fine print". *Nature Phys*, *11*(4), 291–293

Aaronson, S. (2019). Quantum supremacy: The gloves are off. Retrieved 2019-10-28, from https://www.scottaaronson.com/blog/?p=4372

Anschuetz, E. R. & Cao, Y. (2019). Realizing quantum Boltzmann machines through eigenstate thermalization. arXiv: 1903.01359 [quant-ph]

Arute, F., Arya, K., Babbush, R., Bacon, D., Bardin, J. C., Barends, R., ... Martinis, J. M. (2019). "Quantum supremacy using a programmable superconducting processor". *Nature*, *574*(7779), 505–510

Benedetti, M., Grant, E., Wossnig, L., & Severini, S. (2019). "Adversarial quantum circuit learning for pure state approximation". *New J. Phys. 21*(4), 043023

Bergholm, V., Izaac, J., Schuld, M., Gogolin, C., & Killoran, N. (2018). PennyLane: Automatic differentiation of hybrid quantum-classical computations. arXiv: 1811.04968 [quant-ph]

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., & Lloyd, S. (2017). "Quantum machine learning". *Nature*, *549*(7671), 195–202

Boixo, S., Isakov, S. V., Smelyanskiy, V. N., Babbush, R., Ding, N., Jiang, Z., ... Neven, H. (2018). "Characterizing quantum supremacy in near-term devices". *Nature Physics*, *14*(6), 595–600

Chen, Z.-Y., Zhou, Q., Xue, C., Yang, X., Guo, G.-C., & Guo, G.-P. (2018). "64-qubit quantum circuit simulation". *Science Bulletin*, *63*(15), 964–971.

Ciresan, D., Meier, U., & Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition*. IEEE. arXiv: 1202.2745 [cs]

Csáji, B. C. (2001). "Approximation with artificial neural networks". *Faculty of Sciences, Etvs Lornd University, Hungary*, *24*, 48.

Decoste, D. & Schölkopf, B. (2002). "Training invariant support vector machines". *Machine Learning*, *46*(1/3), 161–190

Farhi, E., Goldstone, J., & Gutmann, S. (2014). A quantum approximate optimization algorithm. arXiv: 1411.4028 [quant-ph]

Farhi, E. & Neven, H. (2018). Classification with quantum neural networks on near term processors. arXiv: 1802.06002 [quant-ph]

Feynman, R. P. (1982). "Simulating physics with computers". *Int J Theor Phys*, *21*(6-7), 467–488

Gao, X., Zhang, Z.-Y., & Duan, L.-M. (2018). "A quantum machine learning algorithm based on generative models". *Sci. Adv. 4*(12), eaat9004. eprint: https://advances.sciencemag.org/content/4/12/eaat9004.full.pdf

Ghobadi, R., Oberoi, J. S., & Zahedinejhad, E. (2019). The power of one qubit in machine learning. arXiv: 1905.01390 [quant-ph]

Gibney, E. (2017). "D-Wave upgrade: How scientists are using the world's most controversial quantum computer". *Nature*, *541*(7638), 447–448

Gidney, C. & Ekerå, M. (2019). How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. arXiv: 1905.09749 [quant-ph]

Giovannetti, V., Lloyd, S., & Maccone, L. (2008). "Quantum random access memory". *Phys. Rev. Lett. 100*(16), 160501

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., ... Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems* (pp. 2672–2680).

Google AI. (2019). Quantum. Retrieved 2019-09-11, from https://ai.google/research/teams/applied-science/quantum/

Grant, E., Benedetti, M., Cao, S., Hallam, A., Lockhart, J., Stojevic, V., ... Severini, S. (2018). "Hierarchical quantum classifiers". *npj Quantum Information*, *4*(1), 65.

Grover, L. K. (1996). "A fast quantum mechanical algorithm for database search". arXiv: quant-ph/9605043

Guerreschi, G. G. & Smelyanskiy, M. (2017). Practical optimization for hybrid quantum-classical algorithms. arXiv: 1701.01450 [quant-ph]

Harrow, A. W., Hassidim, A., & Lloyd, S. (2009). "Quantum algorithm for linear systems of equations". *Phys. Rev. Lett. 103*(15), 150502

Havlíček, V., Córcoles, A. D., Temme, K., Harrow, A. W., Kandala, A., Chow, J. M., & Gambetta, J. M. (2019). "Supervised learning with quantum-enhanced feature spaces". *Nature*, *567*(7747), 209–212

Hilbert, M. & Lopez, P. (2011). "The world's technological capacity to store, communicate, and compute information". *Science*, *332*(6025), 60–65

Ho, J. & Ermon, S. (2016). Generative adversarial imitation learning. In *Advances in neural information processing systems* (pp. 4565–4573).

IBM. (2019). Quantum computing. Retrieved 2019-09-11, from https://www.ibm.com/quantum-computing/

Intel. (2019). Quantum computing. Retrieved 2019-09-11, from https://www.intel.com/content/www/us/en/research/quantum-computing.html

Kania, E. B. & Costello, J. K. (2017). Quantum technologies, U.S.-China strategic competition, and future dynamics of cyber stability. In *2017 International Conference on Cyber Conflict (CyCon U.S.)* IEEE

LeCun, Y. & Cortes, C. (2010). "MNIST handwritten digit database". Retrieved from http://yann.lecun.com/exdb/mnist/

Lee, J.-S., Bang, J., Hong, S., Lee, C., Seol, K. H., Lee, J., & Lee, K.-G. (2019). "Experimental demonstration of quantum learning speedup with classical input data". *Phys. Rev. A*, *99*(1), 012313

Lloyd, S., Mohseni, M., & Rebentrost, P. (2013). Quantum algorithms for supervised and unsupervised machine learning. arXiv: 1307.0411 [quant-ph]

Martín-López, E., Laing, A., Lawson, T., Alvarez, R., Zhou, X.-Q., & O'Brien, J. L. (2012). "Experimental realization of Shor's quantum factoring algorithm using qubit recycling". *Nature Photon*, *6*(11), 773–776

McClean, J. R., Boixo, S., Smelyanskiy, V. N., Babbush, R., & Neven, H. (2018). "Barren plateaus in quantum neural network training landscapes". *Nature communications*, *9*(1), 4812.

McClean, J. R., Romero, J., Babbush, R., & Aspuru-Guzik, A. (2016). "The theory of variational hybrid quantum-classical algorithms". *New J. Phys. 18*(2), 023023

Microsoft. (2019). Quantum computing. Retrieved 2019-09-11, from https://www.microsoft.com/en-us/quantum/

Mitarai, K., Negoro, M., Kitagawa, M., & Fujii, K. (2018). "Quantum circuit learning". *Phys. Rev. A*, *98*(3), 032309

Nielsen, M. A. & Chuang, I. L. (2009). *Quantum computation and quantum information: 10th anniversary edition* (10th). New York, NY, USA: Cambridge University Press

Otterbach, J. S., Manenti, R., Alidoust, N., Bestwick, A., Block, M., Bloom, B., ... Rigetti, C. (2017). Unsupervised machine learning on a hybrid quantum computer. arXiv: 1712.05771 [quant-ph]

Pednault, E., Gunnels, J. A., Nannicini, G., Horesh, L., & Wisnieff, R. (2019). Leveraging secondary storage to simulate deep 54-qubit Sycamore circuits. arXiv: 1910.09534 [quant-ph]

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). "Scikit-learn: Machine learning in Python". *Journal of Machine Learning Research*, *12*, 2825–2830.

Pérez-Salinas, A., Cervera-Lierta, A., Gil-Fuster, E., & Latorre, J. I. (2019). Data re-uploading for a universal quantum classifier. arXiv: 1907.02085 [quant-ph]

Peruzzo, A., McClean, J., Shadbolt, P., Yung, M.-H., Zhou, X.-Q., Love, P. J., ... O'Brien, J. L. (2014). "A variational eigenvalue solver on a photonic quantum processor". *Nat Commun*, *5*(1), 4213

Pradhan, S. S., Ward, W. H., Hacioglu, K., Martin, J. H., & Jurafsky, D. (2004). Shallow semantic parsing using support vector machines. In *Proceedings of the human language technology conference of the North American chapter of the association for computational linguistics: HLT-NAACL 2004* (pp. 233–240).

Preskill, J. (2012). "Quantum computing and the entanglement frontier". arXiv: 1203.5813 [quant-ph]

Preskill, J. (2018). "Quantum computing in the NISQ era and beyond". *Quantum*, *2*, 79

Qulacs. (2019). Qulacs: Variational quantum circuit simulator for quantum computation research. Retrieved from http://qulacs.org/

Raymer, M. G. & Monroe, C. (2019). "The US national quantum initiative". *Quantum Sci. Technol. 4*(2), 020504

Rebentrost, P., Mohseni, M., & Lloyd, S. (2014). "Quantum support vector machine for big data classification". *Phys. Rev. Lett. 113*(13), 130503

Romero, J. & Aspuru-Guzik, A. (2019). Variational quantum generators: Generative adversarial quantum machine learning for continuous distributions. arXiv: 1901.00848 [quant-ph]

Ronnow, T. F., Wang, Z., Job, J., Boixo, S., Isakov, S. V., Wecker, D., ... Troyer, M. (2014). "Defining and detecting quantum speedup". *Science*, *345*(6195), 420–424

Schuld, M., Bocharov, A., Svore, K., & Wiebe, N. (2018). Circuit-centric quantum classifiers. arXiv: 1804.00633 [quant-ph]

Schuld, M. & Killoran, N. (2019). "Quantum machine learning in feature Hilbert spaces". *Phys. Rev. Lett.* *122*(4), 040504

Schuld, M., Sinayskiy, I., & Petruccione, F. (2014). "An introduction to quantum machine learning". *Contemp. Phys. 56*(2), 172–185

Schuld, M., Sinayskiy, I., & Petruccione, F. (2016). "Prediction by linear regression on a quantum computer". *Physical Review A*, *94*(2)

Shin, S. W., Smith, G., Smolin, J. A., & Vazirani, U. (2014). How "quantum" is the D-Wave machine? arXiv: 1401.7087 [quant-ph]

Shor, P. W. (1999). "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer". *SIAM Rev. 41*(2), 303–332

Stokes, J., Izaac, J., Killoran, N., & Carleo, G. (2019). Quantum natural gradient. arXiv: 1909.02108 [quant-ph]

SURFsara. (2019). SURFsara Systems. Retrieved 2019-10-28, from https://userinfo.surfsara.nl/systems/

Sweke, R., Wilde, F., Meyer, J., Schuld, M., Fährmann, P. K., Meynard-Piganeau, B., & Eisert, J. (2019). Stochastic gradient descent for hybrid quantum-classical optimization. arXiv: 1910.01155 [quant-ph]

Verdon, G., Broughton, M., & Biamonte, J. (2017). A quantum algorithm to train neural networks using low-depth circuits. arXiv: 1712.05304 [quant-ph]

Vidal, J. G. & Theis, D. O. (2019). Input redundancy for parameterized quantum circuits. arXiv: 1901.11434 [quant-ph]

Wang, G. (2017). "Quantum algorithm for linear regression". *Physical Review A*, *96*(1)

Yoo, S., Bang, J., Lee, C., & Lee, J. (2014). "A quantum speedup in machine learning: Finding an N-bit boolean function for a classification". *New J. Phys. 16*(10), 103014

Yu, C.-H., Gao, F., & Wen, Q.-Y. (2017). An improved quantum algorithm for ridge regression. arXiv: 1707.09524 [quant-ph]

Appendices

A Classification Results on Other Data

We showed that our QNN classifier is capable of classifying handwritten digits. Here, we apply the QNN described in Section 6.1.3 to more data sets to see its performance on other problems. With this, we aim to show that the classifier is able to adapt to a broad variety of problems. We test our QNN on three different data sets, generated by the scikit-learn (Pedregosa et al., 2011) functions `make_moons`, `make_circles` and `make_classification` with a noise factor of 0.125. The data sets consist of 300 entries, of which 75% was used as training data and 25% as testing data. A data point is two-dimensional with values in $[-1.5, 1.5]$ and has a binary label 0 or 1. It is encoded into a quantum state by single-qubit rotations instead of amplitude encoding. Because of the low dimensional data, we redundantly encode it once in an extra qubit, resulting in a two-qubit classifier. A data point $(x_1 \quad x_2)$ is encoded in the two qubits as $\prod_{i=1}^{n} R_x^{(i)}(x_1) R_y^{(i)}(x_2)$. We also use the data re-uploading technique described in Section 6.1.4 to achieve generalization with just two qubits.

The results are plotted in Figure 11. In all cases, the QNN approximates the function well. We can see that increasing the number of layers allows the network to learn more complex shapes, but even with 2 layers it finds the desired shape. Note that due to the encoding method and the non-linear behavior of rotation gates, the QNN has an easy time finding wavelike patterns in data.

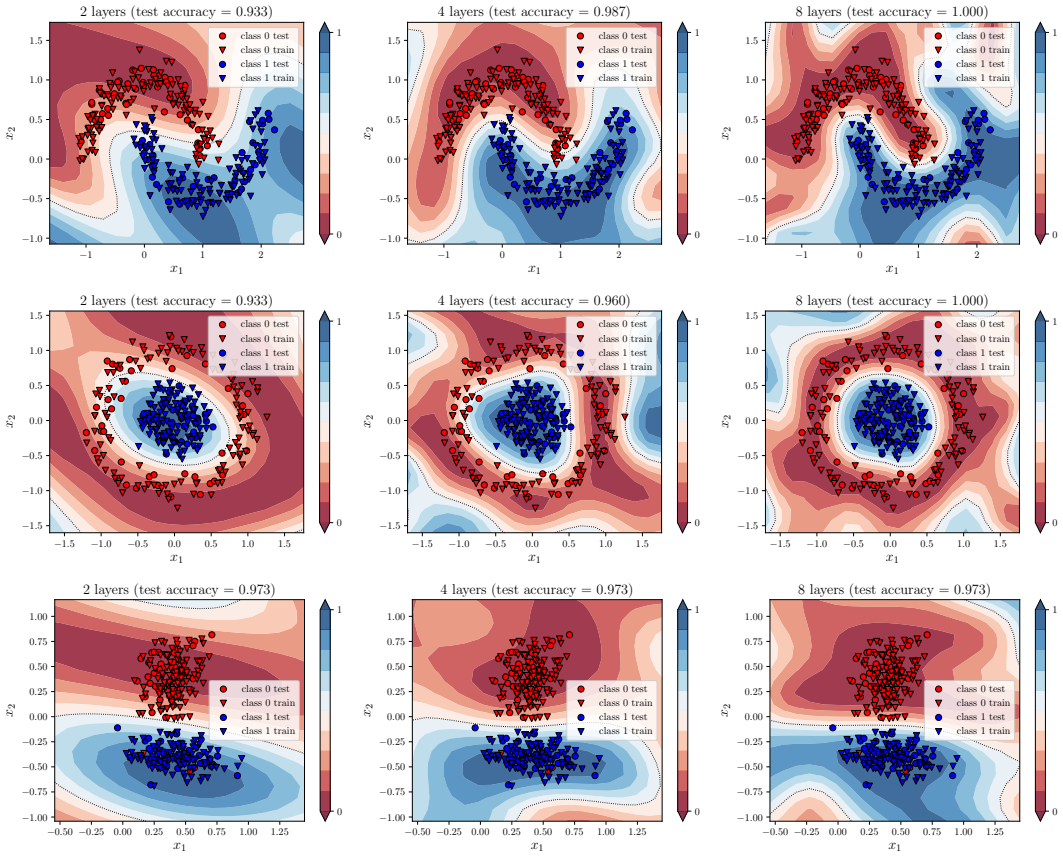


Figure 11: Classification results of a two-qubit QNN on the `make_moons` (top row), `make_circles` (middle row) and `make_classification` (bottom row) data sets. All networks were trained for 10 epochs using RMSprop as optimizer with learning rate 0.05 and momentum 0.5. The red and blue contours represent the certainty of classifying 0 and 1 respectively.

B CPU and GPU Simulation Benchmarks

Quantum circuit simulation is known to be exponentially hard on classical computers. The state space of a quantum system grows exponentially with the number of particles, making simulation of higher amount of particles intractable on a classical computer. To make our simulations as efficient as possible, we compare the performance of the Qulacs simulator on CPU and GPU. We run the benchmarks on the Lisa system of SURFsara. The node used has Intel Xeon Gold 5118 CPUs and an NVIDIA TITAN RTX GPU.

We measure how the execution time increases as the amount of qubits increases. The measure we test is 50 executions of 8 layers of the circuit described in Figure 6. The parameters are randomly generated from a uniform distribution. We also measure $\langle Z \rangle$ every iteration and compare the results of the GPU simulation with the CPU simulation to make sure they agree. The results are visualized in Figure 12. We can see that increasing the number of cores improves simulation times for higher number of qubits. For qubit counts from 18 and higher, GPU simulation is much more efficient. If we look at Table 2, we see that single-core CPU simulation is most efficient up to 12 qubits. While GPU always seems to beat CPU with 24 cores, running on a large number of cores is a good alternative if there is no GPU available.

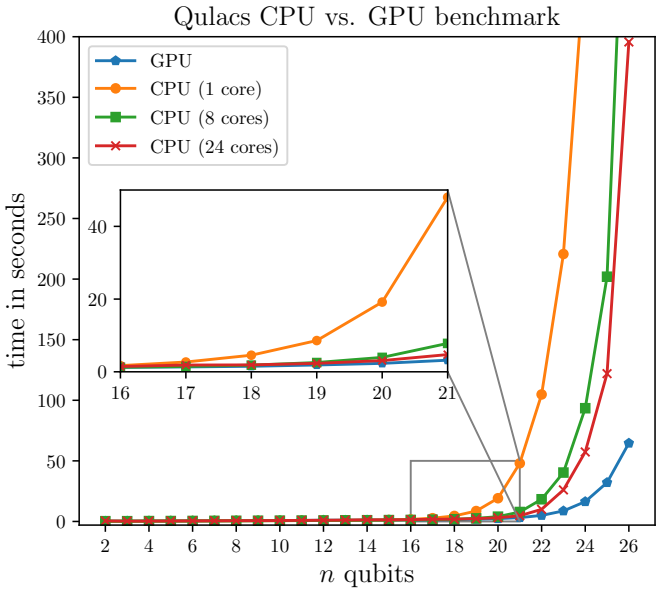


Figure 12: Comparison of quantum simulation runtime between CPU and GPU.

n qubits	GPU	CPU (1 core)	CPU (8 cores)	CPU (24 cores)
2	0.201s	**0.143s**	0.223s	0.466s
4	0.353s	**0.278s**	0.310s	0.320s
6	0.465s	**0.376s**	0.408s	0.479s
8	0.612s	**0.511s**	0.549s	0.664s
10	0.758s	**0.636s**	0.677s	0.787s
12	1.039s	**0.784s**	0.837s	1.021s
14	1.064s	1.024s	**1.016s**	1.369s
16	1.244s	1.695s	**1.242s**	1.463s
18	**1.518s**	4.558s	1.830s	1.895s
20	**2.319s**	19.176s	3.961s	3.081s
22	**4.983s**	104.811s	18.318s	10.014s
24	**16.353s**	466.285s	93.447s	57.340s
26	**64.594s**	2058.394s	634.003s	395.665s

Table 2: Qulacs quantum circuit simulator benchmark results. The fastest result per n qubits is shown in bold.